功能：生成曼宁G2三次几何样条闭曲线
输入参数：m_aVertex-数据点，为受保护成员。
输出参数：BezierPoint-样条曲线分段贝齐尔表示的B样条控制顶点，为受保护成员。

```
void CreatManningSpline()
{
	double lenth,dotx,doty;
	int n=GetVertexCount();
	int n0=n-2;
	Alpha.SetSize(n);
	CArray<CArray<double,double>,CArray<double,double>&> NewAlpha;
	NewAlpha.SetSize(n);
	left.SetSize(n);  right.SetSize(n);
	BezierPoint.SetSize(3*n0+1);
	for(int i=0;i<=n0+1;i++)
	{
		Alpha[i].SetSize(2);
		NewAlpha[i].SetSize(2);
	}
	if(n0==2)
	{

lenth=sqrt((m_aVertex[1].x-m_aVertex[0].x)*(m_aVertex[1].x-m_aVertex[0].x)+(m_aVertex[1].y-m_aVertex[0].y
)*(m_aVertex[1].y-m_aVertex[0].y));
		Alpha[0][0]=-(m_aVertex[1].y-m_aVertex[0].y)/lenth;
		Alpha[0][1]=(m_aVertex[1].x-m_aVertex[0].x)/lenth;
		Alpha[1][0]=(m_aVertex[1].y-m_aVertex[0].y)/lenth;
		Alpha[1][1]=-(m_aVertex[1].x-m_aVertex[0].x)/lenth;
		Alpha[2][0]=Alpha[0][0];
		Alpha[2][1]=Alpha[0][1];
		right[0]=2.*lenth;
		left[0]=right[0];
		right[1]=right[0];
		left[1]=right[0];
		right[2]=right[0];
		left[2]=right[0];
	}
	if(n0>2)
	{
		for(i=1;i<=n0;i++)
		{

lenth=sqrt((m_aVertex[i+1].x-m_aVertex[i-1].x)*(m_aVertex[i+1].x-m_aVertex[i-1].x)+(m_aVertex[i+1].y-m_aV
ertex[i-1].y)*(m_aVertex[i+1].y-m_aVertex[i-1].y));
			Alpha[i][0]=(m_aVertex[i+1].x-m_aVertex[i-1].x)/lenth;
			Alpha[i][1]=(m_aVertex[i+1].y-m_aVertex[i-1].y)/lenth;
		}
		Alpha[0][0]=Alpha[n0][0];
		Alpha[0][1]=Alpha[n0][1];
		Alpha[n0+1][0]=Alpha[1][0];
		Alpha[n0+1][1]=Alpha[1][1];
		for(i=1;i<=n0;i++)
		{

lenth=sqrt((m_aVertex[i+1].x-m_aVertex[i].x)*(m_aVertex[i+1].x-m_aVertex[i].x)+(m_aVertex[i+1].y-m_aVerte
x[i].y)*(m_aVertex[i+1].y-m_aVertex[i].y));
			dotx=(2.*Alpha[i][0]+Alpha[i+1][0])/3.*(m_aVertex[i+1].x-m_aVertex[i].x)/lenth;
			doty=(2.*Alpha[i][1]+Alpha[i+1][1])/3.*(m_aVertex[i+1].y-m_aVertex[i].y)/lenth;
			left[i+1]=2.*lenth/(1.+dotx+doty);
			dotx=(2.*Alpha[i+1][0]+Alpha[i][0])/3.*(m_aVertex[i+1].x-m_aVertex[i].x)/lenth;
			doty=(2.*Alpha[i+1][1]+Alpha[i][1])/3.*(m_aVertex[i+1].y-m_aVertex[i].y)/lenth;
			right[i]=2.*lenth/(1.+dotx+doty);
		}
		left[0]=left[n0];
		left[1]=left[n0+1];
		right[0]=right[n0];
		right[n0+1]=right[1];
		for(i=1;i<=n0;i++)
		{

NewAlpha[i][0]=3.*(right[i]*right[i]*(m_aVertex[i].x-m_aVertex[i-1].x)+left[i]*left[i]*(m_aVertex[i+1].x-
m_aVertex[i].x))-right[i-1]*right[i]*right[i]*Alpha[i-1][0]-left[i]*left[i]*left[i+1]*Alpha[i+1][0];

NewAlpha[i][1]=3.*(right[i]*right[i]*(m_aVertex[i].y-m_aVertex[i-1].y)+left[i]*left[i]*(m_aVertex[i+1].y-
m_aVertex[i].y))-right[i-1]*right[i]*right[i]*Alpha[i-1][1]-left[i]*left[i]*left[i+1]*Alpha[i+1][1];
			lenth=sqrt(NewAlpha[i][0]*NewAlpha[i][0]+
```

```
                        NewAlpha[i][1]*NewAlpha[i][1]);
                        NewAlpha[i][0]=NewAlpha[i][0]/lenth;
                        NewAlpha[i][1]=NewAlpha[i][1]/lenth;
                }
                NewAlpha[0][0]=NewAlpha[n0][0];
                NewAlpha[0][1]=NewAlpha[n0][1];
                NewAlpha[n0+1][0]=NewAlpha[1][0];
                NewAlpha[n0+1][1]=NewAlpha[1][1];
                for(i=0;i<=n0-1;i++)
                {

while(fabs(NewAlpha[i][0]-Alpha[i][0])>=0.00001||fabs(NewAlpha[i][1]-Alpha[i][1])>=0.00001)
                        {
                                for(int j=i;j<=n0;j++)
                                {
                                        Alpha[j][0]=NewAlpha[j][0];
                                        Alpha[j][1]=NewAlpha[j][1];
                                }
                                Alpha[0][0]=NewAlpha[n0][0];
                                Alpha[0][1]=NewAlpha[n0][1];
                                Alpha[n0+1][0]=NewAlpha[1][0];
                                Alpha[n0+1][1]=NewAlpha[1][1];
                                for(int k=1;k<=n0;k++)
                                {

lenth=sqrt((m_aVertex[k+1].x-m_aVertex[k].x)*(m_aVertex[k+1].x-m_aVertex[k].x)+(m_aVertex[k+1].y-m_aVerte
x[k].y)*(m_aVertex[k+1].y-m_aVertex[k].y))

dotx=(2.*Alpha[k][0]+Alpha[k+1][0])/3.*(m_aVertex[k+1].x-m_aVertex[k].x)/lenth;

doty=(2.*Alpha[k][1]+Alpha[k+1][1])/3.*(m_aVertex[k+1].y-m_aVertex[k].y)/lenth;
                                        left[k+1]=2.*lenth/(1.+dotx+doty);


dotx=(2.*Alpha[k+1][0]+Alpha[k][0])*(m_aVertex[k+1].x-m_aVertex[k].x)/lenth;

doty=(2.*Alpha[k+1][1]+Alpha[k][1])*(m_aVertex[k+1].y-m_aVertex[k].y)/lenth;
                                        right[k]=2.*lenth/(1.+dotx+doty);
                                }
                                left[0]=left[n0];
                                left[n0+1]=left[1];
                                right[n0+1]=right[1];
                                for(int l=1;l<=n0;l++)
                                {

NewAlpha[l][0]=3.*(right[l]*right[l]*(m_aVertex[l].x-m_aVertex[l-1].x)+left[l]*left[l]*(m_aVertex[l+1].x-
m_aVertex[l].x))-right[l-1]*right[l]*right[l]*Alpha[l-1][0]-left[l]*left[l]*left[l+1]*Alpha[l+1][0];

NewAlpha[l][1]=3.*(right[l]*right[l]*(m_aVertex[l].y-m_aVertex[l-1].y)+left[l]*left[l]*(m_aVertex[l+1].y-
m_aVertex[l].y))-right[l-1]*right[l]*right[l]*Alpha[l-1][1]-left[l]*left[l]*left[l+1]*Alpha[l+1][1];

lenth=sqrt(NewAlpha[l][0]*NewAlpha[l][0]+NewAlpha[l][1]*NewAlpha[l][1]);
                                        NewAlpha[l][0]=NewAlpha[l][0]/lenth;
                                        NewAlpha[l][1]=NewAlpha[l][1]/lenth;
                                }
                        }
                }
        }
        for(i=0;i<n0;i++)
        {
                BezierPoint[3*i].x=m_aVertex[i].x;
                BezierPoint[3*i].y=m_aVertex[i].y;
                BezierPoint[3*i+1].x=int(m_aVertex[i].x+right[i]*Alpha[i][0]/3.);
                BezierPoint[3*i+1].y=int(m_aVertex[i].y+right[i]*Alpha[i][1]/3.);
                BezierPoint[3*i+2].x=int(m_aVertex[i+1].x-left[i+1]*Alpha[i+1][0]/3.);
                BezierPoint[3*i+2].y=int(m_aVertex[i+1].y-left[i+1]*Alpha[i+1][1]/3.);
        }
        BezierPoint[3*n0].x=m_aVertex[n0].x;
        BezierPoint[3*n0].y=m_aVertex[n0].y;
}
```